



Client SDK und User-Exits



Manuel Tromm

DOPaK 2018

Client SDK – Einleitung

Worum geht es beim Client SDK?

- Zugriff & Manipulation der D5 WebClient GUI
 - Berechnungen & Formularlogik
 - Constraints & Plausibilität
 - Layout
- nicht direkt auf Server realisierbare Funktionalität
- User-Exits, TableData, Gentable und Gadgets

Hilfe

- Client SDK API Dokumentation
- HowTos und programmierte Beispiele

Documents Client SDK

Search Documents Client SDK

- exitRegistry
- gentableRegistry
- GridModel
- DocumentsContext

Methods

- closeDialog
- closeDialog
- encodeURL
- executeScript
- getBaseParams
- getBaseParamsQueryString
- getExtendedSearchContext
- getFormContext
- getDialogContext
- getGentableContext
- getI18nContext
- getLocalStorage
- getSessionStorage
- getURL
- getWebContext
- openConfirmationDialog
- openExtendedSearchView
- openGridView
- openGridView
- openGridView
- openGridView
- openMessageDialog
- openOuter
- openTableDialog
- openTablePanel
- updateGridView

ExtendedSearchContext

- ExtendedSearchFormModel
- FieldModel
- FileContext
- FileFieldModel
- FileFormModel
- FormModel
- GadgetContext
- GadgetForm
- GentableContext
- GentableGridColumnModel
- GentableGridModel
- GentableGridRowModel
- I18nContext
- LocalStorage
- Messages

DocumentsContext

Constructors

The universal interface for user-exits, gentable and gadgets. It includes opening of dialogs and navigations to folders, files, the extended search etc. Additional interfaces are available via the `UserContext`, `FileContext`, `GadgetContext` and the `GentableContext`.

Since: 5.0a

Methods

openTableDialog(title, url, options)

Opens a dedicated dialog for the tabledata plugin.

Basically, there are two different dialog types available, one (internal) frame dialog (default) and another (external) popup window dialog. The internal frame dialog appears like an ordinary DOCUMENTS 5 dialog, the external popup window dialog is the more traditional way. This function supports fully automatic setting of the field data, gentable row data or email dialog recipient data on a tabledata row selection. Alternatively, a custom handler function can be implemented which will be executed on tabledata row selection.

If the automatic field setting should be used, the `exitOptions` option must be specified. Otherwise, if a custom row selection handler should be used, the `onSelect` option must be specified.

If both options are set by accident, `onSelect` option overrides the `exitOptions` option and also any automatic data setting.

See the examples below for recommended implementations of `detail.jsp`. Legacy D5 implementations of `detail.jsp` are fully supported so updating the code is not mandatory in most cases.

Name	Type	Description
<code>title</code>	String	the title of the dialog window
<code>url</code>	String	the url of the <code>table.jsp</code>

options	Name	Type	Description
	<code>onSelect</code>	function	custom handler function that is executed after the selection of a tabledata row entry, the selected data is available as a function parameter. Notice: if this option is set, none of the fields will be set automatically.
	<code>width</code>	Number	the width of the dialog, default: 400
	<code>height</code>	Number	the height of the dialog, default: 300
	<code>sortByColumn</code>	String	the column that sorts the tabledata, add '+' or '-' at the end of the column name to set the sort order. Default: '+'
	<code>searchText</code>	String	the default search text, tabledata is searched on opening
	<code>searchColumns</code>	String	the columns to search the searchText in

Since: 5.0c

Examples

```

**Custom tabledata row selection handler**
documents_ctx.exitRegistry.registerFileFieldExitCallback("crsContext", "crsname", function(documentsContext, options) {
  var title = "sasprocPartner";
  var url = documentsContext.encodeURL("./rest/js/tabledata/table.jsp" + "?" + documentsContext.getBaseParamsQueryString());

  documentsContext.openTableDialog(title, url, {
    width: 400,
    height: 300,
    sortByColumn: "sasprocPartner", // optional
    onSelect: function(data) {
      documentsContext.openMessageDialog("", 3000, stringify(data));
      console.log(data);
    }
  });
}

```

Client SDK – Rückblick

Callbacks

- implementieren eigene Funktionalität (JavaScript)
- bewirken Änderung des Standardverhaltens je Situation
- haben vollen Zugriff auf Client SDK API
- werden von D5 WebClient bei bestimmten Ereignissen ausgeführt

Registrierung

- verknüpft Callback mit Ereignis
- erfolgt mit Funktionen der `exitRegistry` oder `gentableRegistry`

Client SDK – Rückblick

DocumentsContext

- repräsentiert universale Schnittstelle zu Client SDK API
- Zugriff auf div. Sub-Contexts (`UserContext`, `FileContext`, `GentableContext` etc.)
- Callbacks werden mit Funktionen aus Contexts implementiert
- verfügbar in User-Exits, Gentable, Gadgets, ...

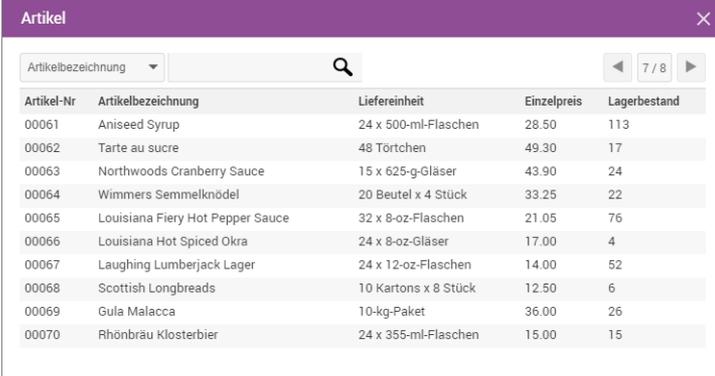
Externe Ressourcen

- externe Verzeichnisstruktur, separiert von D5 Installation
- Lagerort für eigenen Custom-Code, AddOns, Bibliotheken, Konfigurationen
- Konfiguration (Mapping) in Tomcat erfolgt mit `documents.xml`
- `script.jsp` importiert Skript-Ressourcen in D5 WebClient

Neues von TableData

Neue Funktion `openTableDataDialog()`

- elementare Vereinfachung zur bisherigen Variante
- Anzeige im aktuellen D5 Dialog-Layout
- automatisches Setzen von Feldern
- Suchspalte, Suchbegriff und Sortierung
- individuelle Parameter konfigurierbar
- automatische Codierung aller Parameter
- in Mappe, Gentable, Erweiterte Suche, Script-Parameter-Dialog, ...
- einfache Migration (`table.jsp` kompatibel, ggf. minimale Anpassung in `detail.jsp`)



The screenshot shows a dialog window titled 'Artikel' with a search bar and a table of items. The table has five columns: 'Artikel-Nr', 'Artikelbezeichnung', 'Liefereinheit', 'Einzelpreis', and 'Lagerbestand'. The data is as follows:

Artikel-Nr	Artikelbezeichnung	Liefereinheit	Einzelpreis	Lagerbestand
00061	Aniseed Syrup	24 x 500-ml-Flaschen	28.50	113
00062	Tarte au sucre	48 Törtchen	49.30	17
00063	Northwoods Cranberry Sauce	15 x 625-g-Gläser	43.90	24
00064	Wimmers Semmelknödel	20 Beutel x 4 Stück	33.25	22
00065	Louisiana Fiery Hot Pepper Sauce	32 x 8-oz-Flaschen	21.05	76
00066	Louisiana Hot Spiced Okra	24 x 8-oz-Gläser	17.00	4
00067	Laughing Lumberjack Lager	24 x 12-oz-Flaschen	14.00	52
00068	Scottish Longbreads	10 Kartons x 8 Stück	12.50	6
00069	Gula Malacca	10-kg-Paket	36.00	26
00070	Rhönbräu Klosterbier	24 x 355-ml-Flaschen	15.00	15

Neues von TableData

Beispiel: `openTableDataDialog()`

Aufruf mit Titel, URL
und Optionen

Dialog intern/extern

Suchspalte, -begriff
und Sortierung

Custom-Parameter

empfangene Daten
aus `detail.jsp`
verarbeiten

```

1  var title = "Artikel",
2      url = documentsContext.encodeURL("./ext/jsp/product/table.jsp" + "...");
3
4  documentsContext.openTableDataDialog(title, url, {
5      width: 800,
6      height: 400,
7      popupWin: false, // default: false
8
9      searchColumn: "Artikelbezeichnung",
10     searchText: "Coffee Arabica",
11
12     sortColumn: "Artikel-Nr",
13
14     params : {
15         supplierId : "86305"
16     },
17
18     onSelect: function(data) {
19         // set selected table row data
20     }
21 });

```

Artikel		
Artikelbezeichnung	Coffee Arabica	<input type="text"/>
Artikel-Nr	Artikelbezeichnung	Liefereinheit
00061	Aniseed Syrup	24 x 500-ml-F
00062	Tarte au sucre	48 Törtchen
00063	Northwoods Cranberry Sauce	15 x 625-g-GL
00064	Wimmers Semmelknödel	20 Beutel x 4
00065	Louisiana Fiery Hot Pepper Sauce	32 x 8-oz-Flas
00066	Louisiana Hot Spiced Okra	24 x 8-oz-Gläs
00067	Laughing Lumberjack Lager	24 x 12-oz-Fla
00068	Scottish Longbreads	10 Kartons x
00069	Gula Malacca	10-kg-Paket
00070	Rhönbräu Klosterbier	24 x 355-ml-F

Exits in erweiterter Suche

Beispiel: TableData

- analog zum Beispiel in Mappe (DOPaK 2016)
- Import von externen Daten in Suchfelder

Umsetzung

- nutzt bekannte JSPs (`table.jsp`, `detail.jsp`)
- Callback 1 bereitet Exit auf Feld vor
- Callback 2 auf Feld öffnet Tabelle

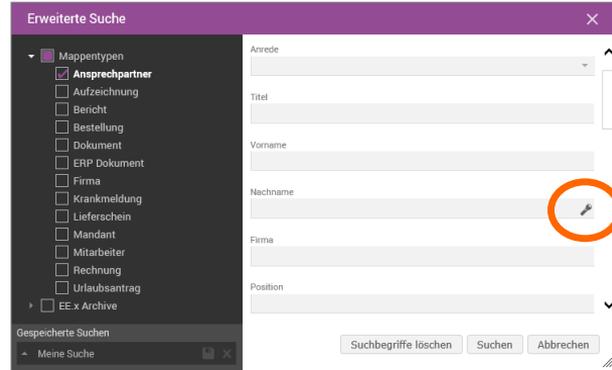
The screenshot shows two overlapping windows from the 'otris software' application. The top window, titled 'Erweiterte Suche', has a dark sidebar with a tree view of 'Mappentypen' (Map Types) including 'Ansprechpartner' (checked), 'Aufzeichnung', 'Bericht', 'Bestellung', 'Dokument', 'ERP Dokument', 'Firma', 'Krankmeldung', 'Lieferschein', 'Mandant', 'Mitarbeiter', 'Rechnung', and 'Urlaubsbewerbung'. The main area contains search fields for 'Anrede', 'Titel', 'Vorname', 'Nachname', 'Firma', and 'Position'. An orange circle highlights a key icon in the search form. An arrow points from this icon to the 'Ansprechpartner' window below, which displays a table of search results.

Firma	Ansprechpartner	PLZ	Ort	Land
Mère Paillarde	Jean Fresnière	H1J 1C3	Montréal	Kanada
Morgenstern Gesundkost	Alexander Feuer	04179	Leipzig	Deutschland
North/South	Simon Crowther	SW7 1RZ	London	Großbritannien
Océano Atlántico Ltda.	Yvonne Moncada	1010	Buenos Aires	Argentinien
Old World Delicatessen	Rene Phillips	99508	Anchorage	USA
Otilies Käseladen	Henriette Pfalzheim	50739	Köln	Deutschland
Paris spécialités	Marie Bertrand	75012	Paris	Frankreich
Pericles Comidas clásicas	Guillermo Fernández	05033	México D.F.	Mexiko
Ficcolo und mehr	Georg Pippis	5020	Salzburg	Österreich
Princesa Isabel Vinhos	Isabel de Castro	1756	Lisboa	Portugal

Exits in erweiterter Suche

Beispiel: TableData (Callback 1)

- Ausführung vor Anzeige der Suchfelder
- untersucht ausgewählte Suchquellen des Baums
- untersucht verfügbare Felder des Formulars
- Feld „Nachname“ erhält ggf. Exit mit Button



Registrierung

Suchquellen / Felder

Mappentyp suchen

Feld suchen

Exit auf Feld setzen

```

1 documents.sdk.exitRegistry.registerSearchExitCallback(
2     "ExtendedSearch.afterSetModelData", function(documentsContext, options) {
3
4     var extSearchContext = documentsContext.getExtendedSearchContext(),
5         searchSources = extSearchContext.getSelectedSearchSourceNames(),
6         fields = extSearchContext.getSearchFormModel().getSearchFields(), field;
7
8     if(searchSources.indexOf("crmContact") >= 0) {
9
10        if(field = fields.get("crmName")) {
11
12            field.setExit({ type: "button", image: "entypo:flashlight", tooltip: "..."});
13        }
14    }
15 });

```

Exits in erweiterter Suche

Beispiel: TableData (Callback 2)

- Ausführung bei Klick auf Exit-Button bei „Nachname“
- öffnet Dialog mit `openTableData()`
- gewählter Eintrag (`detail.jsp`) wird in Feld übertragen

Nachname

Suchfeld

Ansprechpartner

Firma	Ansprechpartner	PLZ	Ort
Mère Paillarde	Jean Fresnière	H1J 1C3	Montréal
Morgenstern Gesundkost	Alexander Feuer	04179	Leipzig
North/South	Simon Crowther	SW7 1RZ	London
Océano Atlántico Ltda.	Yvonne Moncada	1010	Buenos Aires
Old World Delicatessen	Rene Phillips	99508	Anchorage
Ottiles Käseladen	Henriette Pfalzheim	50739	Köln
Paris spécialités	Marie Bertrand	75012	Paris
Pericles Comidas Clasicas	Guillermo Fernández	05033	México D.F.
Piccolo und mehr	Georg Pippis	5020	Salzburg
Princesa Isabel Vinhos	Isabel de Castro	1756	Lisboa

Nachname: Bertrand

Registrierung

Feld „Nachname“

TableData-Dialog

Feldwert setzen

```

20 documents.sdk.exitRegistry.registerSearchFieldExitCallback(
21     "crmName", function(documentsContext, options) {
22
23     var title = "Ansprechpartner",
24         url = documentsContext.encodeURL("./ext/jsp/tabledata/table.jsp" + "..."),
25         field = options.searchField;
26
27     documentsContext.openTableDataDialog(title, url, {
28         width: 800,
29         height: 400,
30         onSelect: function(data) {
31             field.setValue(data.crmName);
32         }
33     });
34 });

```

detail.jsp

```

crmAccountNumber: "PARIS",
crmName: "Bertrand",
crmFirstName: "Marie",
...

```

Exit-Callbacks in erweiterter Suche

- Callbacks bei Ereignissen in Feldern (z.B. `focusout`, `change`, `button`, ...)
 - `exitRegistry.registerSearchFieldExitCallback(fieldName, fn)`
- Callbacks bei Ereignissen im Dialog
 - `exitRegistry.registerSearchExitCallback(event, fn)`

Beispiele für Callback-Ereignisse im Dialog	Beschreibung	Abbruch bei return false;
<code>ExtendedSearch.afterSetModelData</code>	nach Empfang der Daten	-
<code>ExtendedSearch.afterRenderSearchForm</code>	nach Rendern der Suchfelder	-
<code>ExtendedSearch.beforeExecuteSearch</code>	vor Start der Suche	ja
<code>ExtendedSearch.afterExecuteSearch</code>	nach Ende der Suche	-
<code>ExtendedSearch.afterToggleSearchSource</code>	nach Wechsel der Suchquelle	-
<code>ExtendedSearch.afterToggleSearchMask</code>	nach Wechsel der Suchmaske	-
<code>ExtendedSearch.afterToggleHitListMask</code>	nach Wechsel der Trefferliste	-

Exits in Skript-Parameter-Dialogen

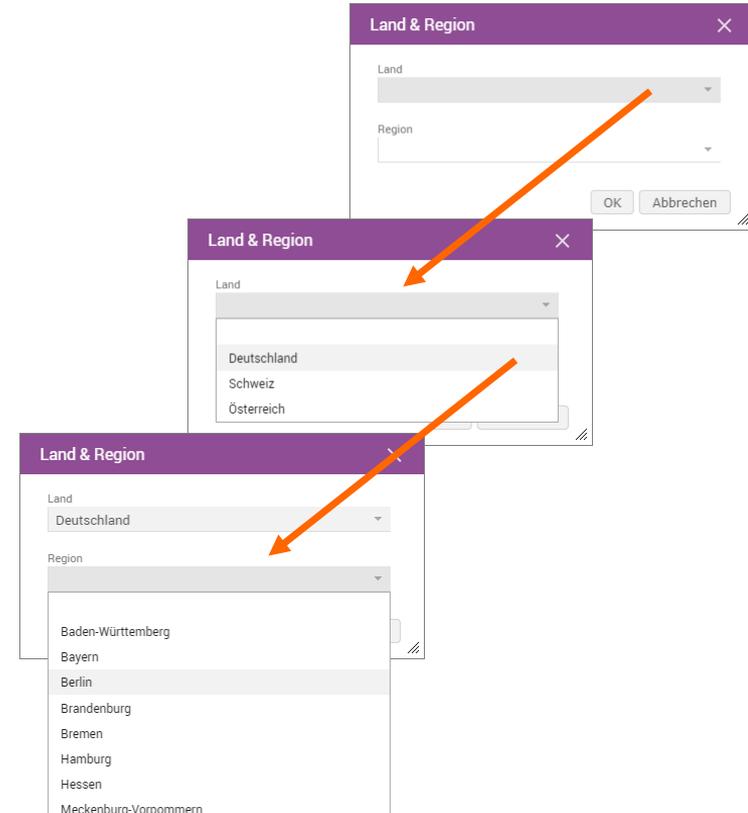
Beispiel: Dynamische DropDown-Liste

Anforderungen

- Feld „Land“ hält Länder
- Feld „Region“ hält Regionen je Land
- Änderung von Land passt Regionen automatisch an
- Feld „Region“ ist gesperrt, falls kein Land gewählt

Umsetzung

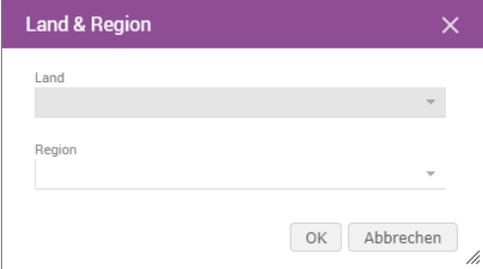
- Callback 1 bei Start des Skripts bereitet Exit vor
- Callback 2 auf Feld „Land“ ruft Skript auf
- Skript auf Server liefert Regionen aus



Exits in Skript-Parameter-Dialogen

Beispiel: Dynamische DropDown-Liste (Callback 1)

- Ausführung vor Anzeige der Felder
- Feld „Land“ erhält Exit bei Wertänderung
- Feld „Region“ ist initial schreibgeschützt



Land & Region

Land

Region

OK Abbrechen

Registrierung

alle Felder

Exit auf Feld „Land“

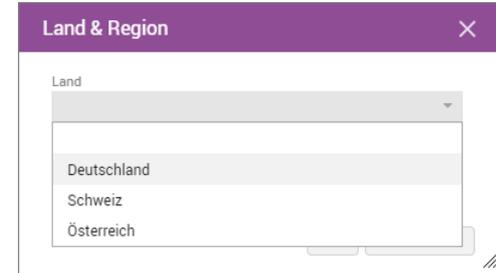
Feld „Region“ sperren

```
1 documents.sdk.exitRegistry.registerScriptParameterExitCallback(  
2     "country-region", "ScriptParameter.afterSetModelData", function(documentsContext, options) {  
3  
4     var fields = options.scriptForm.getFields();  
5  
6     var field1 = fields.get("crmCountry");  
7     field1.setExit({ type: "change" });  
8  
9     var field2 = fields.get("crmRegion");  
10    field2.setGuiReadOnly(true);  
11 });
```

Exits in Skript-Parameter-Dialogen

Beispiel: Dynamische DropDown-Liste (Callback 2)

- Ausführung bei Änderung von Land
- Skript-Aufruf (synchron) mit Feldwert aus „Land“
- setzt aus Skript empfangene Werte in Feld „Region“



Registrierung

Feld „Land“ auslesen

Script mit Parameter
country ausführen

Feld „Region“ setzen

```

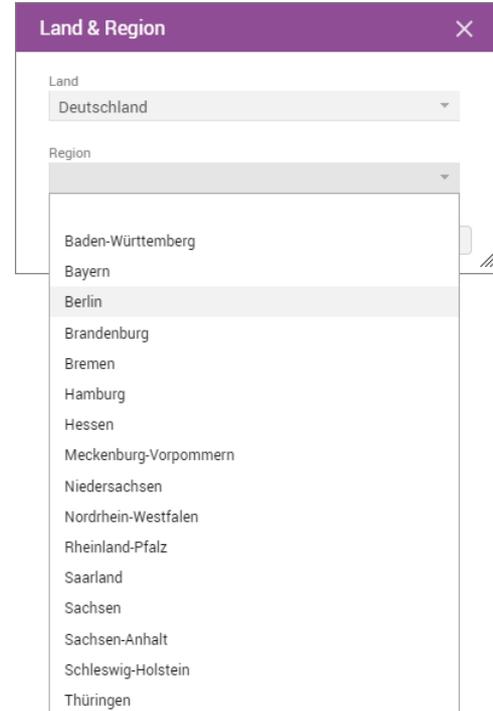
15 documents.sdk.exitRegistry.registerScriptParameterFieldExitCallback(
16     "country-region", "crmCountry", function(documentsContext, options) {
17
18     var fields = options.scriptForm.getFields(),
19         country = options.scriptFieldValue;
20
21     var regions = documentsContext.executeScript("countryScript", { "country" : country });
22
23     var field = fields.get("crmRegion");
24     field.setEnumValues(regions);
25     field.setGuiReadOnly(regions === "");
26 });
  
```

Exits in Skript-Parameter-Dialogen

Beispiel: Dynamische DropDown-Liste (Skript)

- `countryScript` gibt Regionen je Land zurück
- liefert keine Regionen, falls kein Land gewählt ist
- Skript-Parameter `country` wird von Skript-Aufruf übermittelt

```
1 var regions = {
2   "Germany": ["Baden-Württemberg", "Bayern", "Berlin", "Brandenburg", "Bremen", "..."],
3   "Austria": ["Burgenland", "Kärnten", "Niederösterreich", "Oberösterreich", "..."],
4   "Switzerland": ["Zürich", "Bern", "Luzern", "Uri", "Schwyz", "Obwalden", "..."]
5 }
6
7 if(!country) {
8   return "";
9 }
10
11 return regions[country];
```



Exit-Callbacks in Skript-Parameter-Dialogen

- Callbacks bei Ereignissen in Feld (z.B. `focusout`, `change`, `button`, ...)
 - `exitRegistry.registerScriptParameterFieldExitCallback(scriptName, fieldName, fn)`
- Callbacks bei Ereignissen in Dialog
 - `exitRegistry.registerScriptParameterExitCallback(scriptName, event, fn)`

Beispiele für Callback-Ereignisse in Dialog	Beschreibung	Abbruch bei return false;
<code>ScriptParameter.afterSetModelData</code>	nach Empfang der Daten	-
<code>ScriptParameter.beforeRender</code>	vor Rendern des Dialogs	-
<code>ScriptParameter.afterRender</code>	nach Rendern des Dialogs	-
<code>ScriptParameter.beforeExecuteScript</code>	vor Start des Skripts	ja

Asynchrone Skripte mit JSON

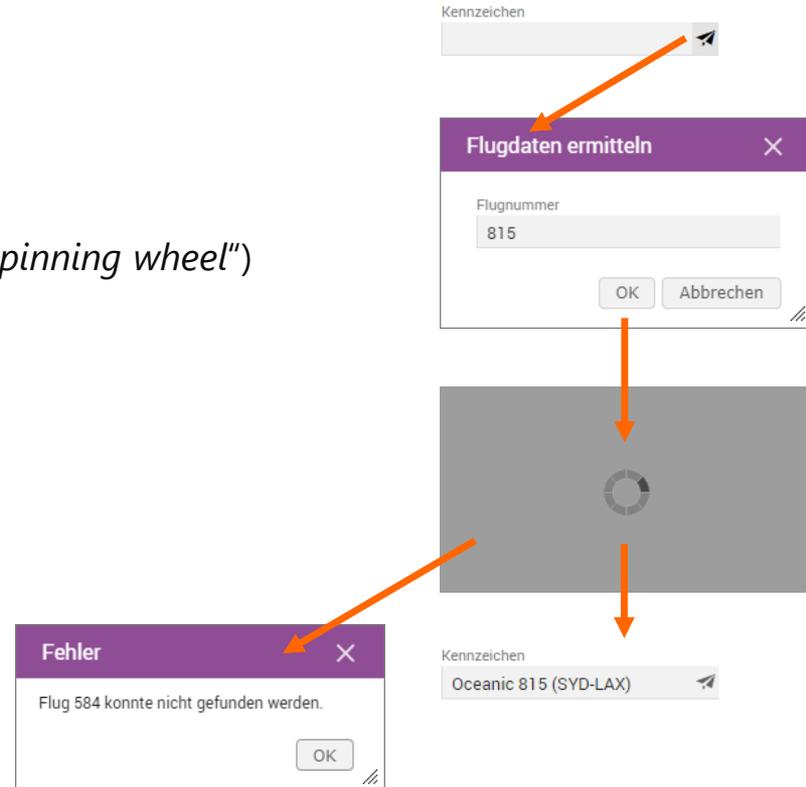
Beispiel: Flugdaten ermitteln

Anforderungen

- Flugdaten mit Flugnummer suchen
- Mappe sperren während laufender Suchanfrage („*spinning wheel*“)
- Flug-Kennzeichen in Feld übertragen, falls OK
- Fehlermeldung, falls nicht gefunden

Lösung

- Callback ruft Skript mit Parameterdialog
- Skript sucht und gibt Flugdatensatz zurück
- Datenformat: JSON



Asynchrone Skripte mit JSON

Beispiel: Flugdaten ermitteln (Skript)

Fall: Fehler

Flugdatensatz
„Oceanic 815“

```

1  if(flightNumber !== "815") {
2      context.errorMessage = "Flug " + flightNumber + " konnte nicht gefunden werden.";
3      return -1;
4  }
5
6  var data = {
7      airline: "Oceanic",
8      number: 815,
9      departure: {
10         IATA: "SYD",
11         time: "2004-09-22 14:55",
12         city: "Sydney"
13     },
14     arrival: {
15         IATA: "LAX",
16         time: "2004-09-23 10:42",
17         city: "Los Angeles"
18     },
19     numbers: [4, 8, 15, 16, 23, 42]
20 }
21
22 return JSON.stringify(data);

```

Fall: OK

Ausgabe (JSON)

```

{"airline":"Oceanic","number":815,"departure":{"IATA":"SYD","time":"2004-09-22 14:55","city":"Sydney"},"ar

```

Skript-Parameter

Name	Bezeichnung	Typ
flightNumber	Flugnummer	String

Asynchrone Skripte mit JSON

Beispiel: Flugdaten ermitteln (Callback)

Registrierung

spinning wheel ein

Skript-Aufruf (async)

Fall: OK

JSON Daten

spinning wheel aus

Fall: Fehler

```
1 documents.sdk.exitRegistry.registerFileFieldExitCallback(  
2     "flightRecord", "flightLabel", function(documentsContext, options) {  
3  
4     var scriptOptions = {  
5         async: true,  
6         useScriptParameterDialog: true  
7     };  
8  
9     documentsContext.startBusyPanel("MainFile");  
10  
11     var promise = documentsContext.executeScript("flight-data", { }, scriptOptions);  
12  
13     promise.then(function(jsonData) {  
14  
15         var data = JSON.parse(jsonData),  
16             field = options.fileForm.getFieldByName("flightLabel"),  
17             value = data.airline + " " + data.number + "...";  
18  
19         field.setValue(value);  
20         documentsContext.stopBusyPanel("MainFile");  
21     })  
22     .catch(function(error) {  
23         documentsContext.stopBusyPanel("MainFile");  
24     });  
25 });
```

Exit-Callbacks in Gentable

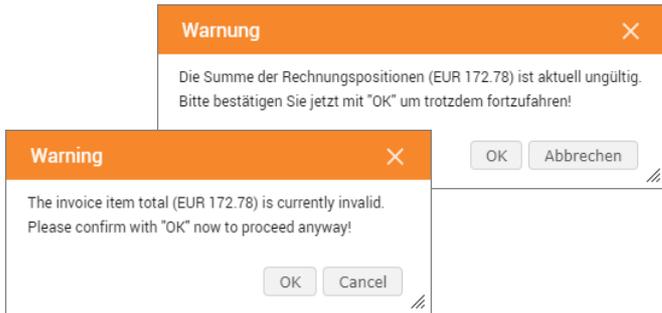
- Callbacks bei Ereignissen in Feld oder Button (z.B. in `<event>` oder `<function>`)
 - `gentableRegistry.registerCallback(gentableDefinitionName, event, fn)`
- Callbacks bei allg. Ereignissen in Gentable
 - `gentableRegistry.registerCallback(gentableDefinitionName, event, fn)`

Beispiele für Callback-Ereignisse in Gentable	Beschreibung	Abbruch bei return false;
<code>Gentable.beforeRender</code>	vor Rendern der Tabelle	-
<code>Gentable.afterRender</code>	nach Rendern der Tabelle	-
<code>Gentable.beforeStore</code>	vor Speichern der Tabelle	ja
<code>Gentable.afterStore</code>	nach Speichern der Tabelle	-
<code>Gentable.afterRowSelect</code>	nach Selektion der Zeile	-

Internationalisierung (I18n)

I18nContext

- einfacher Zugriff auf eigene Nachrichten in aktueller Sprache des Benutzers
- lokalisierte Bezeichner in eigenen Properties-Dateien je Sprache (z.B. `custom_de.properties`, `custom_en.properties`, ...)
- Unterstützung für Wildcards
- Lagerort: Externe Ressource (z.B. `/ext/resource/` oder `/ext/i18n/`)



```
1 var messages = documentsContext.getI18nContext().getMessages("custom"),
2   title = messages.get("title"),
3   description = messages.get("description", null, ["172.78"]);
4
5 documentsContext.openConfirmationDialog(title, description, onOK, onCancel);
```

```
25 title = Warnung
26 description = Die Summe der Rechnungspositionen (EUR {1}) ist aktuell ungültig...
```

```
25 title = Warning
26 description = The invoice item total (EUR {1}) is currently invalid...
```

Internationalisierung (I18n)

TableData

- lokalisierte Bezeichner in Properties-Dateien je Sprache
(TableData_de.properties, TableData_en.properties, ...)
- Lagerort: Externe Ressource (z.B. /ext/resource/ oder /ext/i18n/)

```

52 <tab:tableconfig id="item" />
53   ...
54   <tab:headline>itemno,itemname,unit,itemprice,stock</tab:headline>
55   ...
56 </tab:tableconfig>

```

Artikel
✕

Artikelbezeichnung

🔍
◀ 3/8 ▶

Artikel-Nr	Artikelbezeichnung	Liefereinheit	Einzelpreis	Lagerbestand
21	Sir Rodney's Scones	24 Packungen x 4 Stück	10.0000	3

Item
✕

item name

🔍
◀ 3/8 ▶

item number	item name	unit	item price	stock
21	Sir Rodney's Scones	24 Packungen x 4 Stück	10.0000	3

```

16 itemno = Artikel-Nr
17 itemname = Artikelbezeichnung
18 unit = Liefereinheit
19 itemprice = Einzelpreis
20 stock = Lagerbestand

```

```

16 itemno = item number
17 itemname = item name
18 unit = unit
19 itemprice = item price
20 stock = stock

```

Internationalisierung (I18n)

Gentable Feldtyp NUMBER

- Daten sind unabhängig von der aktuellen Sprache
- sprachabhängige Anzeige von Dezimal- und Tausendertrennzeichen
- Formatierung der Nachkommastellen für die Anzeige
- Dezimaltrennzeichen und Präzision der Nachkommastellen für die Daten
- Berechnungen in Exits immer in JavaScript-Datentyp `Number`

<input type="checkbox"/>	Nr	Artikelnr.	Artikeltext	Menge	Stückpreis	Betrag	KST	Konto
<input type="checkbox"/>	1	83212408	BMW Engine Oil LL04 5W30	12	166,60	1.999,20		

<input type="checkbox"/>	no	item number	description	amount	unit price	total	cost unit	account
<input type="checkbox"/>	1	83212408	BMW Engine Oil LL04 5W30	12	166.60	1,999.20		

Internationalisierung (I18n)

Gentable Feldtyp NUMBER

```
<field number="5">
  <label>de:Betrag;en:total</label>
  <title>total</title>
  <type>NUMBER</type>
  <width>80</width>
  <numberFormat>n.2</numberFormat>
  <editable>true</editable>
</field>
```

Betrag	1.999,20
total	1,999.20

Dezimaltrenner
(Daten & Anzeige)

Nachkommastellen

Tausendertrenner

Präzision (Daten)

Daten-XML

Listen-Export Outbars Documents (Anzeige) Locale/Format Log-Buch Lizenzen Eigenschaften

Format-Einstellungen
 Datumformat TT.MM.## Dezimaltrennzeichen .
 Mehrsprachigkeit unterstützen

Locale 1
 Locale de Datumformat TT.MM.## Dezimaltrennzeichen .

Locale 2
 Locale en Datumformat MM.TT.## Dezimaltrennzeichen .

Bezeichnung	Wert
groupingNumeric	1
groupingNumeric_de	.
groupingNumeric_en	.
precisionNumeric	4

```
<table><tr>...<td title="total">1999,2000</td>...</tr></table>
```

Vielen Dank!

Manuel Tromm

www.otris.de

otris software AG
Königswall 21
44137 Dortmund